

Improved Approximation for Metric Multi-Cover

Santanu Bhowmick* Tanmay Inamdar† Kasturi Varadarajan‡

Department of Computer Science
University of Iowa, Iowa City, USA

Abstract

We consider the metric multi-cover problem (MMC). The input consists of two point sets Y (servers) and X (clients) in an arbitrary metric space $(X \cup Y, d)$, a positive integer k that represents the coverage demand of each client, and a constant $\alpha \geq 1$. Each server can have a single ball of arbitrary radius centered on it. Each client $x \in X$ needs to be covered by at least k such balls centered on servers. The objective function that we wish to minimize is the sum of the α -th powers of the radii of the balls.

Bar-Yehuda and Rawitz [4] gave a $3^\alpha \cdot k$ -approximation for the MMC problem. Bhowmick et al. [5, 6] showed that an $O(1)$ approximation (independent of the coverage demand k) exists for the special case in which X, Y are points in a fixed-dimensional space \mathbb{R}^d . In this paper, we present an $O(1)$ -approximation for the MMC problem in any arbitrary metric space, improving the result of Bar-Yehuda and Rawitz [4] and matching the guarantee for fixed-dimensional Euclidean spaces in [5, 6]. We also obtain an $O(1)$ for the non-uniform version of the MMC, where clients have possibly different coverage requirements.

*santanu-bhowmick@uiowa.edu

†tanmay-inamdar@uiowa.edu

‡kasturi-varadarajan@uiowa.edu

1 Introduction

We consider the metric multi-cover problem (MMC) in this paper. The input to this problem consists of two point sets Y (servers) and X (clients) in an arbitrary metric space $(X \cup Y, d)$, a positive integer k that represents the coverage demand of each client, and a constant $\alpha \geq 1$.

Consider an assignment $r : Y \rightarrow \mathbb{R}^+$ of *radii* to each server in Y . We say that X is j -covered under r (or by r) if for each $x \in X$, there are at least j servers $y \in Y$ such that the ball of radius $r(y)$ centered at y contains x . That is, X is j -covered if for each $x \in X$,

$$|\{y \in Y \mid d(x, y) \leq r(y)\}| \geq j.$$

The *cost* of assignment r is defined to be $\text{cost}(r) = \sum_{y \in Y} (r(y))^\alpha$.

Any assignment $r : Y \rightarrow \mathbb{R}^+$ that k -covers X is a feasible solution to the MMC problem. The objective to be minimized is the cost $\sum_{y \in Y} (r(y))^\alpha$. We assume that $k \leq |Y|$, for otherwise there is no feasible solution. The problem is NP-hard in general, as we point out below, and we are interested in approximation algorithms for the problem that run in polynomial time.

1.1 Related Work

We review some of the previous works which considered the MMC problem and its variants. A version of the MMC problem arises naturally in fault-tolerant wireless sensor network design. The clients (X) and the servers (Y) are points in the plane, and the requirement is to construct disks centered at the servers such that each client is covered by at least k distinct server disks. A server disk corresponds to the area covered by some wireless antenna placed at the server, whose power consumption is proportional to the area being serviced by the antenna. The objective function is to minimize the power consumption of the antennas placed at the servers while meeting the coverage requirement of each client. This problem is thus a special case of the MMC problem, with X, Y in the plane and $\alpha = 2$. This special case has been studied in several recent works [1, 5, 6].

Abu-Affash et al. [1] considered the special case $\alpha = 2$ of the MMC problem where X and Y are subsets of \mathbb{R}^2 and the metric is the Euclidean distance. They gave an $O(k)$ approximation for the problem. (Throughout the paper, it is implicit that we only refer to polynomial time algorithms.) Following their work, Bhowmick et al. [5] gave an $O(1)$ approximation, thus obtaining a guarantee that is independent of the coverage demand. Their approximation guarantee also holds for a *non-uniform* generalization of the MMC problem where each client can have an arbitrary demand. The algorithm in [5] was further generalized in the journal article [6], in which an approximation guarantee of $4 \cdot (27\sqrt{2})^\alpha$ was achieved for the MMC problem in the plane, for any $\alpha \geq 1$.

In \mathbb{R}^d , their approximation guarantee is $(2d) \cdot (27\sqrt{d})^\alpha$, which depends on the dimension. Motivated by this, Bhowmick et al. [6] ask whether an $O(1)$ guarantee is possible in any metric space. This is the question considered in this article. The metric space setting generalizes not only the Euclidean distance in any dimension, but also the shortest path distance amidst polygonal obstacles in \mathbb{R}^2 or \mathbb{R}^3 , and in graphs.

Bar-Yehuda and Rawitz [4] were the first to give an approximation algorithm for the MMC problem in which X, Y are points in an arbitrary metric space. They presented a $3^\alpha \cdot k$ approximation guarantee, using the local-ratio technique. They also consider the non-uniform version of the problem, where the coverage demand of each client is an arbitrary integer that is not necessarily related to the demands of the other clients. They obtain a $3^\alpha \cdot k_{\max}$ approximation for this version, where k_{\max} is now the maximum client demand.

The case $k = 1$ for the MMC problem is a traditional covering problem, and has a much longer history. In \mathbb{R}^d for any fixed d , a polynomial-time approximation scheme (PTAS) using dynamic programming exists, as shown by Bilò et al. [7] (improving on the work of Lev-Tov and

Peleg [11] who obtained a PTAS for the plane and $\alpha = 1$). For the general metric setting (with $k = 1$), the primal-dual method has been successful in obtaining constant factor approximations, as demonstrated in [8, 10].

The MMC problem is known to be NP-hard even when X, Y are point sets in the plane and $k = 1$, for any $\alpha > 1$. This was established by Bilò et al. [7] for $\alpha \geq 2$, and subsequently for $\alpha > 1$ by Alt et al. [2].

Some recent results that involve geometric set multi-covering problems can be found in [9, 3]. Reducing to a set multi-covering problem does not seem to be an effective way to deal with the MMC, partly because in a feasible solution to the MMC each server can contribute only one ball. This issue is discussed in greater detail in [6].

1.2 Our Results

In this paper, we present an $O(1)$ approximation for the MMC problem in any metric space, achieving a guarantee that is independent of the coverage demand k . This resolves a problem left open by Bhowmick et al. [6]. To be more precise, our approximation guarantee is $2 \cdot (108)^\alpha$. We have not attempted to optimize the constants, as our focus is on answering the question of whether a guarantee independent of k is possible.

This result represents a major advance over [6] in our understanding of the MMC problem. To explain this, we first describe the overall approach of [6]. They first compute covers ρ^i for X , for $1 \leq i \leq k$. Each ρ^i is actually a special type of cover called a level i *outer cover*, as described precisely in Section 2. They show that $\sum_{i=1}^k \text{cost}(\rho^i)$ is, up to a constant factor, a lower bound on the cost of the optimal solution to the MMC. Note that a server in Y may contribute a ball to many of the ρ^i , so the union of the ρ^i is not a feasible solution to the MMC.

Their algorithm for computing a k -cover is recursive – it first computes a $(k - 1)$ -cover and then extends it to a k -cover. They bound the increase in cost in going from an $(i - 1)$ -cover to an i -cover by $c_d \cdot \text{cost}(\rho^i)$, where c_d is a constant that depends on the dimension d and α . To extend their approach to the metric setting, one would have to bound the increase in cost in going from an $(i - 1)$ -cover to an i -cover by $c \cdot \text{cost}(\rho^i)$, where c is a constant (that depends only on α).

This is not possible, as the following “high-dimensional” example shows. Suppose that k , the coverage requirement, is even and $\alpha = 1$. Let

$$X = Y = \{\pm e_j \mid 1 \leq j \leq k/2\},$$

where e_j is the point in $\mathbb{R}^{k/2}$ with 1 in the j -th coordinate and 0 in the other coordinates. Thus, each of the k points is both a client and a server; each client has $k - 2$ points at distance $\sqrt{2}$ from it, and one ‘antipodal’ point at distance 2 from it. Here, we can let ρ^i , the level i outer cover, for each i , to be the singleton set $\{\delta(e_1, 2)\}$, where $\delta(p, r)$ denotes the ball of radius r centered at p .

Now suppose the $(k - 1)$ -cover we have at hand is $\{\delta(y, \sqrt{2}) \mid y \in Y\}$. That is, the radius assigned to each $y \in Y$ is $\sqrt{2}$. Now an optimal k -cover is $\{\delta(y, 2) \mid y \in Y\}$, since in any k -cover each client needs to be contained in the ball centered at its antipodal server. Thus the increase in cost incurred by the algorithm in going from the $(k - 1)$ -cover to a k -cover is at least $(2 - \sqrt{2})k$, which is larger than $\text{cost}(\rho^k) = 2$ by a multiplicative factor of $\Omega(k)$. Thus, the curse of dimensionality afflicts the analysis framework of [6].

In this article, we take a different approach. We extract k *disjoint* subsets Y_1, Y_2, \dots, Y_k of Y . For each subset Y_i , we compute a set of balls centered at Y_i that covers X , using as a black-box any constant factor approximation algorithm for 1-covering [8, 10]. This gives us k covers that result in a feasible solution for the MMC problem, as the k server subsets are disjoint.

To obtain an approximation guarantee using this approach, we require that the disjoint subsets Y_1, Y_2, \dots, Y_k satisfy the following property: for each client from some representative

family, Y_i should contain at least one server from among the i servers nearest to the client. This concept appears to be of interest beyond its application to the MMC problem. Notice that the property becomes stricter as i gets smaller. A bad choice of servers Y_i can mean that there is no feasible choice of Y_j for $j < i$. Thus it is not clear that the desired family of k subsets exists. One of our main technical contributions is a constructive proof that a family satisfying some closely related property does indeed exist.

Having constructed the k disjoint subsets, we bound the approximation factor by first showing that the cost of an optimal cover of X using servers Y_i is within a constant of the cost of any i -level outer cover. This is straightforward given the property that Y_i satisfies. To finish, we use the result of [6], that there exists an i -level outer cover ρ^i for each $1 \leq i \leq k$ such that sum of the costs of these k outer covers serves as a lower bound for the optimal MMC solution. Notice that unlike [6], we use the notion of outer covers only in the analysis and not the algorithm itself.

Our approach is robust enough to generalize to the non-uniform version of the MMC problem, where each client has a possibly different demand. We obtain an $O(1)$ approximation for this problem as well. The approach tracks that of the uniform case very closely, but needs one additional idea to overcome a technical complication. We describe the generalization to the non-uniform MMC in [Appendix B](#).

Our algorithm for the MMC is given in [Section 3](#), after establishing needed preliminaries in [Section 2](#).

2 Preliminaries

Let $\delta(p, r)$ denote the ball of radius r centered at p , i.e., $\delta(p, r) = \{u \in X \cup Y \mid d(p, u) \leq r\}$. For brevity, we slightly abuse the notation and write $\delta(p, d(p, q))$ as $\delta(p, q)$. The *cost* of a set B of balls, denoted $\text{cost}(B)$, is defined to be the sum of the α -th powers of the radii of the balls.

Any assignment $r : Y \rightarrow \mathbb{R}^+$ corresponds to the set of balls $\{\delta(y, r(y)) \mid y \in Y\}$. Note that the cost of assignment r is the same as the cost of the corresponding set of balls. Instead of saying that r j -covers X , we will often say that the corresponding set of balls j -covers X . We will say that a set of balls covers X instead of saying it 1-covers X .

We first fix an arbitrary total ordering $<_Y$ of the servers in Y . For each $x \in X$ and $1 \leq j \leq |Y|$, we define $y^j(x)$ to be the j -th closest point in Y to x using distance d . Ties are broken using the total ordering $<_Y$. For any $x \in X$, we define the i -neighborhood ball of a client x as $\delta(x, y^i(x))$. We define the i -neighborhood of x , $YN_x(i)$, as $\{y^j(x) \mid 1 \leq j \leq i\}$.

2.1 Computing 1-covers

We will need as a black-box an algorithm that, given subsets $X' \subseteq X$ and $Y' \subseteq Y$, computes a 1-cover of X' using servers in Y' . That is, the algorithm must return an assignment of radii $r : Y' \rightarrow \mathbb{R}^+$ such that each client $x \in X'$ is contained in at least one ball centered on a server in Y' . Computing a 1-cover of minimum cost is thus the special case for the MMC problem where $k = 1$. As mentioned in [Section 1.1](#), even this version is NP-hard, but it does admit constant-factor approximations [8, 4]. Let $\text{Cover}(X', Y', \alpha)$ denote an algorithm that returns a 1-cover of X' using servers in Y' with cost at most 3^α times the cost of an optimal 1-cover of X' using servers in Y' .

2.2 Outer Cover

Our work also relies on the notion of an *outer cover*, which is described in Bhowmick et al. [6]. We adopt the definition of an outer cover from [6] as follows:

Definition 2.1. Given point sets X, Y in a metric space $(X \cup Y, d)$, positive integer i and $\alpha \geq 1$, an outer cover of level i is an assignment $\rho^i : Y \rightarrow \mathbb{R}^+$ of radii to the servers such that for each client $x \in X$, there is a server $y \in Y$ such that

1. The ball $\delta(y, \rho^i(y))$ contains x i.e. $d(y, x) \leq \rho^i(y)$.
2. Radius of the ball at y is large, that is, $\rho^i(y) \geq d(x, y^i(x))$

Given a level i outer cover ρ^i , and a client $x \in X$, any server y that satisfies the two conditions in the definition above is said to *serve* x ; we also say that the corresponding ball $\delta(y, \rho^i(y))$ serves x .

To appreciate why outer covers play an important role, consider any k -cover of the set X of clients. Fix $1 \leq i \leq k$. Form a set B of balls by adding, for each client in X , the i -th largest ball in the k -cover that covers the client. The set B thus constructed is seen to be a level i outer cover.

The sum of the costs of the optimal i -th level outer covers, for $1 \leq i \leq k$, gives a lower bound on the cost of the optimal solution to the MMC. This is stated precisely in the theorem below. The proof can be found in [6], but for completeness, it is extracted and given in [Appendix A](#).

Theorem 1. Let $r' : Y \rightarrow \mathbb{R}^+$ be any assignment that constitutes a feasible solution to the MMC problem. For each $1 \leq i \leq k$, let μ_i denote the cost of an optimal outer cover of level i . Then

$$\sum_{i=1}^k \mu_i \leq 3^\alpha \cdot \text{cost}(r').$$

3 Solving the MMC Problem

In this section, we describe a constant factor approximation for the MMC problem. Recall that our input consists of two point sets Y (servers) and X (clients) in an arbitrary metric space $(X \cup Y, d)$, a positive integer k that represents the coverage demand of each client, and the constant $\alpha \geq 1$. Our algorithm first computes a family \mathcal{F} consisting of k pairwise disjoint subsets of Y . It then invokes $\text{Cover}(X, Y', \alpha)$ for each $Y' \in \mathcal{F}$ to obtain k covers of X . Because server subsets in \mathcal{F} are disjoint, these k covers yield a k -cover of X . Before describing the algorithm in detail, we introduce needed concepts.

An (α, β) -ruling set of a graph $G = (V, E)$ is a set $S \subseteq V$ such that every path in G between any two vertices in S has at least α edges in it, and for every $u \in V \setminus S$, there exists a vertex $v \in S$ such that u is reachable from v using a path in G having at most β edges.

Let $G_i = (X, E_i)$ be the intersection graph of i -neighborhoods of X i.e. $(x, x') \in E_i$ if $YN_x(i) \cap YN_{x'}(i) \neq \emptyset$. We note that for any G_i, G_j such that $l \leq i < j \leq k$, G_i is a sub-graph of G_j since the i -neighborhood of any client is contained within its j -neighborhood.

Claim 3.1. There is a polynomial time algorithm that, given X, Y , and k , computes a hierarchy

$$X_k \subseteq X_{k-1} \subseteq \cdots \subseteq X_{l-1} \subseteq X_1,$$

where each $X_i \subseteq X$ is a $(3, 2)$ ruling set of G_i .

Proof. Henceforth, we refer to any $(3, 2)$ ruling set as simply a ruling set. Given a ruling set X_i of G_i , we describe how to compute a ruling set X_{i-1} of G_{i-1} such that $X_i \subseteq X_{i-1}$. Since G_{i-1} is a subgraph of G_i , we have that the (hop) distance in G_{i-1} between any two vertices in X_i is at least 3. We initialize X_{i-1} with X_i and assume that all vertices in G_{i-1} are initially unmarked. We repeat the following process till G_{i-1} does not contain any unmarked vertices: mark all vertices in G_{i-1} within distance 2 of X_{i-1} , and then add an arbitrary unmarked vertex from G_{i-1} to X_{i-1} .

We can construct the hierarchy of ruling sets by starting with an arbitrary ruling set X_k of the graph G_k , and then constructing the successive ruling sets in the hierarchy by the process described above. \square

3.1 Computing k Disjoint Covers

Our algorithm for the MMC problem is described in [Algorithm 1](#). We begin by setting a parameter l to be $\lceil k/2 \rceil$. We then use [Claim 3.1](#) to compute a hierarchy of ruling sets $X_k \subseteq X_{k-1} \subseteq \dots \subseteq X_l$, truncating it at l . Any client that belongs to $\bigcup_{i=l}^k X_i$ is termed as a *ruling client*. For each client x , we denote the l -neighborhood $YN_x(l)$ by YP_x , the *private servers* of x .

Algorithm 1 MetricMultiCover(X, Y, k, α)

```

1: For each  $y \in Y$ , assign  $r(y) \leftarrow 0$  and mark  $y$  as available.
2:  $l \leftarrow \lceil k/2 \rceil$ 
3: Compute  $X_k \subseteq X_{k-1} \subseteq \dots \subseteq X_l$  using Claim 3.1.
4: for  $i = k$  to  $l$  do
5:   Let  $Y_i^s \leftarrow \emptyset, Y_i^p \leftarrow \emptyset$ .
6:   for all  $x_c \in X_i$  do
7:     if  $i > l$  then
8:        $y_s \leftarrow$  farthest available server in  $YN_{x_c}(i)$ .
9:        $Y_i^s \leftarrow Y_i^s \cup \{y_s\}$ . Mark  $y_s$  as not available.
10:    if  $i > l$  or ( $i = l$  and  $k$  is odd) then
11:       $y_p \leftarrow$  any available server in  $YP_{x_c}$ .
12:       $Y_i^p \leftarrow Y_i^p \cup \{y_p\}$ . Mark  $y_p$  as not available.
13: for  $i = k$  to  $l + 1$  do
14:    $r(Y_i^s) \leftarrow \text{Cover}(X, Y_i^s, \alpha)$ .
15:    $r(Y_i^p) \leftarrow \text{Cover}(X, Y_i^p, \alpha)$ .
16: if  $k$  is odd then
17:    $r(Y_l^p) \leftarrow \text{Cover}(X, Y_l^p, \alpha)$ .
18: return The assignment  $r : Y \rightarrow \mathbb{R}^+$ 

```

A family \mathcal{F} of disjoint subsets of Y is computed in [Lines 4 to 12](#) of [Algorithm 1](#) – the for loop, whose index i goes down from k to l . In each iteration $i \geq l + 1$, we extract two disjoint sets of servers Y_i^p and Y_i^s , and if k is odd, we extract one server set Y_l^p in iteration l . Notice that when summed over all i from k to l , we get k disjoint server sets. We let \mathcal{F} denote the family consisting of these server subsets.

Observe that in iteration i , we go through each client in $x_c \in X_i$, and use a carefully designed rule to pick two available servers from the i -neighborhood $YN_{x_c}(i)$ of x_c to add to Y_i^p and Y_i^s . These added servers are immediately made unavailable. The fact that X_i is a ruling set in G_i is useful in controlling the impact on server availability for later iterations of the algorithm. The subsequent section is devoted to establishing the crucial fact that such available servers can be found in iteration i .

Once \mathcal{F} is computed, the remainder of the algorithm ([Lines 13 to 18](#)) consists of invoking $\text{Cover}(X, Y', \alpha)$ for each $Y' \in \mathcal{F}$, and returning the union of the k sets of balls. This is clearly a k -cover of X .

Assuming that servers are available whenever the algorithm looks for them, it is evident that for each $x_c \in X_i$, there is (at least) one server in Y_i^p (resp. Y_i^s) that belongs to the i -neighborhood $YN_{x_c}(i)$. We use this property in [Section 3.3](#) to bound the cost of the 1-cover returned by $\text{Cover}(X, Y_i^p, \alpha)$ (resp. $\text{Cover}(X, Y_i^s, \alpha)$) and establish an approximation guarantee.

Remark 1. *It is natural at this point to wonder if it is possible to compute a sequence of disjoint server subsets Z_k, \dots, Z_1 such that for each $1 \leq i \leq k$ and $x_c \in X_i$, Z_i contains at least one server from $YN_{x_c}(i)$. It is easy to see that this is not possible by considering the example instance in [Section 1](#). For a different perspective, suppose that x and x' are two clients such that $YN_x(k) \cap YN_{x'}(k) \neq \emptyset$, and $x \in X_k$. Thus, $x' \notin X_k$. Suppose we pick just one server from $YN_x(k)$, namely $y^k(x)$, and add it to Z_k . When we move to iteration $k-1$, x' may have lost two servers in the transition, namely, $y^k(x)$ and $y^k(x')$, while being “covered” just once. This motivates our approach of picking two servers instead of one in iteration k . The first server we pick is $y^k(x)$, whereas the second server is from YP_x . What we show is that this second server is not a loss for x' , viewed from the perspective of the later iteration when x' shows up in the ruling set. Effectively, then, x' still only loses at most two servers – $y^k(x)$ and $y^k(x')$. But now it has been covered twice.*

In the remainder of this section, we state some straightforward properties concerning the hierarchy of ruling sets $X_k \subseteq X_{k-1} \subseteq \dots \subseteq X_l$. These will be needed in the subsequent section on server availability.

Claim 3.2. *Let x, x' be two distinct clients in X_i . Then $YN_x(i) \cap YN_{x'}(i) = \emptyset$.*

Proof. Since X_i is a ruling set for G_i , any path between x and x' in G_i has at least three edges. Recall that the condition $YN_x(i) \cap YN_{x'}(i) \neq \emptyset$ is equivalent to (x, x') being an edge in G_i . \square

Claim 3.3. *Let $x \in X \setminus X_i$. Then there is at most one $x' \in X_i$ such that $YN_x(i) \cap YN_{x'}(i) \neq \emptyset$.*

Proof. If there are two clients x_1 and x_2 in X_i such that $YN_x(i) \cap YN_{x_1}(i) \neq \emptyset$ and $YN_x(i) \cap YN_{x_2}(i) \neq \emptyset$, then there is a path in G_i with at most two edges connecting x_1 and x_2 . Since the clients x_1 and x_2 belong to X_i , this would contradict the fact that X_i is a $(3, 2)$ ruling set. \square

Claim 3.4. *Let $x_i \in X_i$ and $x_j \in X_j$ be any two distinct clients for $l \leq i < j \leq k$. Then, $YN_{x_i}(i) \cap YN_{x_j}(l) = \emptyset$.*

Proof. Since $i < j$, we have $X_j \subseteq X_i$ and hence the clients x_i and x_j both belong to the ruling set X_i , implying that $YN_{x_i}(i) \cap YN_{x_j}(i) = \emptyset$. Since $l \leq i$, the claim follows, as $YN_{x_j}(l)$, the l -neighborhood of x_j , is contained in $YN_{x_j}(i)$. \square

3.2 Server Availability

Fix an iteration i of the for loop in [Line 4](#) in [Algorithm 1](#). In such an iteration, the algorithm considers each $x_c \in X_i$ in the inner for loop in [Line 6](#). For each x_c , it looks for up to two available servers within $YN_{x_c}(i)$ and uses them. In order for the algorithm to be correct, such available servers must exist when the algorithm looks for them. In this section, which is the core of our analysis, we show that this is indeed the case.

We introduce some notation for this purpose. For $x \in X$, let $A_x(i)$ denote the set of available servers within $YN_x(i) = \{y^j(x) \mid 1 \leq j \leq i\}$ at the *beginning* of iteration i . Thus, $|A_x(k)| = k$. Furthermore, $A_x(i-1) \subseteq A_x(i)$ for $l+1 \leq i \leq k$. Obviously, $A_x(i) \subseteq YN_x(i)$.

The *threshold level* of a ruling client x (denoted by $\text{th}(x)$) is defined as:

$$\forall x \in \bigcup_{i=l}^k X_i, \quad \text{th}(x) = \begin{cases} k, & \text{if } x \in X_k \\ j, & \text{if } x \in X_j \setminus X_{j+1}, \quad l \leq j < k \end{cases}$$

The threshold level of x denotes the iteration of the outer loop of the algorithm in which client x first enters the ruling set. In any iteration $k \geq j \geq \text{th}(x) + 1$, the client x can lose neighboring servers because of the server choices made by (the algorithm for) other clients, i.e., clients in

the ruling set X_j . On the other hand, for $l + 1 \leq j \leq \text{th}(x)$, x is itself part of the ruling set X_j . In these iterations, it can only lose neighboring servers because of its own server choices. The next two claims address these two phases.

We now show that any ruling client x has enough available servers in its $\text{th}(x)$ neighborhood at the iteration $i = \text{th}(x)$ of the outer loop of [Algorithm 1](#).

Claim 3.5. *Let x be any ruling client, and let $i = \text{th}(x)$. Then*

- (a) $|A_x(i) \cap YP_x| \geq l - (k - i)$.
- (b) $|A_x(i)| \geq 2i - k = k - 2(k - i)$.

Proof. We look at the servers chosen during iteration j of the outer loop, for $i < j \leq k$. Consider any client $x_j \in X_j$. By [Claim 3.4](#), $YN_x(i)$ does not intersect the l -neighborhood ball $YN_{x_j}(l)$. Hence, during the execution of [Line 11](#) in the inner for loop corresponding to x_j , no server is made unavailable from $YN_x(i)$. This is because the server chosen in [Line 11](#) belongs to $YN_{x_j}(l) = YP_{x_j}$.

Thus, during iteration j , servers from $YN_x(i)$ can become unavailable only during the execution of [Line 8](#) of the inner for loop. We note that by [Claim 3.3](#), there is at most one client $x_j \in X_j$ such that $YN_{x_j}(j) \cap YN_x(j) \neq \emptyset$. Thus, at most one server from $YN_x(i)$ is made unavailable in iteration j .

We conclude that across the $k - i$ iterations before iteration i , there can be at most $k - i$ servers from $YN_x(i)$ that have been made unavailable. Hence, $|A_x(i)| \geq i - (k - i) = 2i - k$. Since $|YN_x(i) \cap YP_x| = l$ and at most $k - i$ servers are made unavailable from the i -neighborhood ball $YN_x(i)$, $|A_x(i) \cap YP_x| \geq l - (k - i)$. \square

For any ruling client x , [Claim 3.5](#) shows that in iteration $i = \text{th}(x)$, when x first enters the ruling set, there are enough available servers in $YN_x(i)$. The following claim aids in asserting this for subsequent iterations, by arguing that in any iteration $i \leq \text{th}(x)$, at most 2 available servers are made unavailable from $YN_x(i)$.

Claim 3.6. *Let x be any ruling client and $l + 1 \leq i \leq \text{th}(x)$. Then*

- (a) $|A_x(i - 1)| \geq |A_x(i)| - 2$
- (b) *If $|A_x(i - 1)| = |A_x(i)| - 2$, then one of the servers in $A_x(i) \setminus A_x(i - 1)$ is the farthest server in $A_x(i)$ from x .*

Proof. Note that $x \in X_i$ since $i \leq \text{th}(x)$. Consider any $x_c \in X_i \setminus \{x\}$. In the iteration of the inner for loop ([Line 6](#)) corresponding to x_c , any servers that are made unavailable belong to $YN_{x_c}(i)$ and are therefore not in $YN_x(i)$, by [Claim 3.2](#) (since $x, x_c \in X_i$). Thus, if any servers in $A_x(i) \subseteq YN_x(i)$ become unavailable in iteration i , then this can happen only in the iteration of the inner for loop corresponding to x . In this iteration of the inner for loop, the servers that become unavailable are y_s , the farthest server from x in $A_x(i)$, and y_p , a different server that is chosen from the available servers in YP_x . Note that $\{y_p, y_s\} \subseteq A_x(i)$. Thus, only the two servers y_s, y_p in $A_x(i)$ become unavailable in iteration i . Furthermore, if $y^i(x) \in A_x(i)$ then $y^i(x)$ is the farthest server in $A_x(i)$ from x , and thus $y^i(x) = y_s$. Thus $A_x(i) \setminus A_x(i - 1) = \{y_s, y_p\}$, and [Claim 3.6 \(a\)](#) holds. Since y_s is the farthest server in $A_x(i)$, [Claim 3.6 \(b\)](#) holds as well. \square

We can now argue that our algorithm always succeeds in finding available servers.

Claim 3.7. *For any $l + 1 \leq i \leq k$, and any $x_c \in X_i$:*

- (a) *There is an available server in $YN_{x_c}(i)$ when the algorithm executes [Line 8](#) in the iteration of the inner for loop ([Line 6](#)) corresponding to x_c .*

- (b) *There is an available server in YP_{x_c} when the algorithm executes **Line 11** in the iteration of the inner for loop (**Line 6**) corresponding to x_c .*

Proof. Since $x_c \in X_i$, we infer that $i \leq \text{th}(x_c)$. Using **Claim 3.6**, we have

$$\begin{aligned}
|A_{x_c}(i)| &\geq |A_{x_c}(i+1)| - 2 \\
&\geq |A_{x_c}(i+2)| - 2 - 2 \\
&\geq \dots \\
&\geq |A_{x_c}(\text{th}(x_c))| - 2 \cdot (\text{th}(x_c) - i) \\
&\geq k - 2(k - i) \quad (\because |A_{x_c}(\text{th}(x_c))| \geq k - 2(k - \text{th}(x_c))) \\
&\geq 2 \quad (\because i \geq l + 1)
\end{aligned}$$

Using an argument from the proof of **Claim 3.6**, none of the servers in $A_{x_c}(i)$ are made unavailable in iteration i till x_c is considered in **Line 6**. Thus, there are at least two servers available when the algorithm executes **Line 8** corresponding to x_c , and **Claim 3.7 (a)** holds.

The argument for **Claim 3.7 (b)** is similar but requires some case analysis. We begin by observing that when the algorithm executes **Line 11** corresponding to x_c , there is at least one available server $y \in YN_{x_c}(i)$. Now suppose that in some iteration $i + 1 \leq j \leq \text{th}(x_c)$, $A_{x_c}(j) \setminus A_{x_c}(j - 1)$ consists of two servers from YP_{x_c} . By **Claim 3.6 (b)**, a server from YP_{x_c} is the farthest server from x_c in $A_{x_c}(j)$. This implies that all servers in $A_{x_c}(j - 1)$ belong to YP_{x_c} , and thus $y \in YP_{x_c}$. This y is available when the algorithm executes **Line 11** corresponding to x_c .

We are left with the case that in each iteration $i + 1 \leq j \leq \text{th}(x_c)$, $A_{x_c}(j) \setminus A_{x_c}(j - 1)$ consists of at most one server from YP_{x_c} . Using **Claim 3.5 (a)**, and the fact that $\text{th}(x_c) - i$ iterations have happened since iteration $\text{th}(x_c)$, we have

$$|A_{x_c}(i) \cap YP_{x_c}| \geq |A_{x_c}(\text{th}(x_c)) \cap YP_{x_c}| - (\text{th}(x_c) - i) \geq l - (k - i) \geq 1.$$

Thus there is at least one server $y' \in A_{x_c}(i) \cap YP_{x_c}$. If the server chosen in **Line 8** corresponding to x_c belongs to YP_{x_c} , then all available servers in $YN_{x_c}(i)$ belong to YP_{x_c} . Thus, once again, some server in YP_{x_c} is available when the algorithm executes **Line 11** corresponding to x_c . If the server chosen in **Line 8** corresponding to x_c does not belong to YP_{x_c} , then $y' \in YP_{x_c}$ is available when the algorithm executes **Line 11** corresponding to x_c . We have thus shown that **Claim 3.7 (b)** holds. \square

If k is even, the algorithm does look for available servers in iteration $i = l$. If k is odd, the algorithm will look for available servers in iteration $i = l$, in **Line 11**. The following claim extends the previous one to handle this. The proof is a straightforward extension.

Claim 3.8. *Suppose k is odd. For iteration $i = l$, and any $x_c \in X_i$, there is an available server in YP_{x_c} when the algorithm executes **Line 11** in the iteration of the inner for loop (**Line 6**) corresponding to x_c .*

3.3 Approximation Guarantee

We have established in **Section 3.2** that **Algorithm 1** successfully computes a family \mathcal{F} of k pairwise disjoint subsets of Y . Let $Y_i \in \mathcal{F}$ be one such subset, where Y_i may be either Y_i^p or Y_i^s . As already observed, Y_i has the property that for any $x \in X_i$, $YN_x(i) \cap Y_i \neq \emptyset$. The following claim uses this property to argue that there is an inexpensive 1-cover of X that only uses servers from Y_i .

Claim 3.9. *Assume that either (a) $l + 1 \leq i \leq k$ and Y_i is either Y_i^p or Y_i^s , or (b) k is odd, $i = l$, and $Y_i = Y_i^p$. Let ρ^i be any outer cover of level i for X using servers from Y . There is a 1-cover of X that uses servers from Y_i and has cost at most $12^\alpha \cdot \text{cost}(\rho^i)$.*

Proof. Consider the set B of balls obtained by expanding each ball in the outer cover ρ^i to 6 times its original radius. We claim

Claim 3.10. *For any client $x \in X$, there is some ball in B that contains x as well as at least one server in Y_i .*

Before proving **Claim 3.10**, we first prove **Claim 3.9** using it. We construct a set B' of balls as follows. Consider any ball $b \in B$. If it does not contain a server from Y_i , we ignore it. If it does contain a server in Y_i , pick an arbitrary such server y , translate b so that it is centered at y , double its radius, and add the resulting ball to B' .

It is possible at this stage that for a server $y \in Y_i$, there are several balls in B' centered at y . From each such concentric family, discard from B' all but the largest of the concentric balls. It follows from **Claim 3.10** that B' covers each client in X . Since each ball in B' is obtained by translating and scaling some ball in the outer cover ρ^i by a factor of 12, the cost of B' is at most $12^\alpha \cdot \text{cost}(\rho^i)$. This establishes **Claim 3.9**.

We now turn to the proof of **Claim 3.10**. From the definition of G_i , we have that for any edge (x', x'') in G_i ,

$$d(x', x'') \leq d(x', y^i(x')) + d(x'', y^i(x'')). \quad (1)$$

Now consider an arbitrary client $x \in X$. Since X_i is a (2,3)-ruling set in G_i , there is a path π in G_i with at most 2 edges (and 3 vertices) that connects x to some vertex $\bar{x} \in X_i$. Let $\delta(y, \rho^i(y))$ be the biggest ball in outer cover ρ^i that serves at least one vertex on path π . Suppose that it serves vertex $\hat{x} \in \pi$. (\hat{x} could be the same as x or \bar{x} .) Using the definition of an outer cover of level i , and the way we pick the ball $\delta(y, \rho^i(y))$, it follows that for any vertex $x' \in \pi$,

$$d(x', y^i(x')) \leq \rho^i(y). \quad (2)$$

Thus,

$$d(y, x) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, x]} d(x', x'') \right) \leq 5\rho^i(y).$$

Here, we denote by $\pi[\hat{x}, x]$ the sub-path of π from \hat{x} to x , and use Inequalities 1 and 2 in the second step.

Now, $YN_{\bar{x}}(i) \cap Y_i \neq \emptyset$. Let $\bar{y} \in YN_{\bar{x}}(i) \cap Y_i$ be chosen arbitrarily. Clearly, $d(\bar{x}, \bar{y}) \leq d(\bar{x}, y^i(\bar{x})) \leq \rho^i(y)$.

We calculate that

$$d(y, \bar{y}) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, \bar{x}]} d(x', x'') \right) + d(\bar{x}, \bar{y}) \leq 6\rho^i(y).$$

Thus, the ball $\delta(y, 6\rho^i(y))$ contains both x and $\bar{y} \in Y_i$, completing the proof of **Claim 3.10**. \square

Remark 2. *With a more detailed argument, the factor 12^α can be improved. For instance, a bound of 11^α is almost immediate from the proof.*

We can now establish the approximation guarantee for **Algorithm 1** and our main result.

Theorem 2. *Given point sets X and Y in a metric space $(X \cup Y, d)$ and a positive integer $k \leq |Y|$, **Algorithm 1** runs in polynomial time and returns a k -cover of X with cost at most $2 \cdot (12 \cdot 9)^\alpha$ times that of an optimal k -cover.*

Proof. It is evident that the algorithm runs in polynomial time and returns a k -cover r . Let r' be any optimal assignment. By [Theorem 1](#), there exist outer covers ρ^i , for $1 \leq i \leq k$, such that

$$\sum_{i=1}^k \text{cost}(\rho^i) \leq 3^\alpha \text{cost}(r').$$

Assume that either (a) $l+1 \leq i \leq k$ and Y_i is either Y_i^p or Y_i^s , or (b) k is odd, $i = l$, and $Y_i = Y_i^p$. From [Claim 3.9](#), we conclude that there is a 1-cover for X that uses servers Y_i and has cost at most $12^\alpha \cdot \text{cost}(\rho^i)$. Since $\text{Cover}(X, Y_i, \alpha)$, which is invoked in [Algorithm 1](#) returns a 3^α approximation, the cost of the 1-cover it returns is at most $(12 \cdot 3)^\alpha \text{cost}(\rho^i)$.

At most two 1-covers are computed for each i , once with server set Y_i^s and once with server set Y_i^p . Thus,

$$\text{cost}(r) \leq 2 \cdot (12 \cdot 3)^\alpha \cdot \sum_{i=l}^k \text{cost}(\rho^i) \leq 2 \cdot (12 \cdot 3)^\alpha \cdot \sum_{i=1}^k \text{cost}(\rho^i) \leq 2 \cdot (12 \cdot 9)^\alpha \cdot \text{cost}(r').$$

□

References

- [1] A. Karim Abu-Affash, Paz Carmi, Matthew J. Katz, and Gila Morgenstern. Multi cover of a polygon minimizing the sum of areas. *Int. J. Comput. Geometry Appl.*, 21(6):685–698, 2011. [1](#)
- [2] Helmut Alt, Esther M. Arkin, Hervé Brönnimann, Jeff Erickson, Sándor P. Fekete, Christian Knauer, Jonathan Lenchner, Joseph S. B. Mitchell, and Kim Whittlesey. Minimum-cost coverage of point sets by disks. In *Symposium on Computational Geometry*, pages 449–458, 2006. [2](#)
- [3] Nikhil Bansal and Kirk Pruhs. Weighted geometric set multi-cover via quasi-uniform sampling. In *ESA*, pages 145–156, 2012. [2](#)
- [4] Reuven Bar-Yehuda and Dror Rawitz. A note on multicovering with disks. *Comput. Geom.*, 46(3):394–399, 2013. [1](#), [3](#)
- [5] Santanu Bhowmick, Kasturi R. Varadarajan, and Shi-Ke Xue. A constant-factor approximation for multi-covering with disks. In *Symposium on Computational Geometry*, pages 243–248, 2013. [1](#)
- [6] Santanu Bhowmick, Kasturi R. Varadarajan, and Shi-Ke Xue. A constant-factor approximation for multi-covering with disks. *JoCG*, 6(1):220–234, 2015. [1](#), [2](#), [3](#), [4](#), [11](#)
- [7] Vittorio Bilò, Ioannis Caragiannis, Christos Kaklamani, and Panagiotis Kanellopoulos. Geometric clustering to minimize the sum of cluster sizes. In *ESA*, pages 460–471, 2005. [1](#), [2](#)
- [8] Moses Charikar and Rina Panigrahy. Clustering to minimize the sum of cluster diameters. *J. Comput. Syst. Sci.*, 68(2):417–441, 2004. [2](#), [3](#)
- [9] Chandra Chekuri, Kenneth L. Clarkson, and Sarel Har-Peled. On the set multicover problem in geometric settings. *ACM Transactions on Algorithms*, 9(1):9, 2012. [2](#)
- [10] Ari Freund and Dror Rawitz. Combinatorial interpretations of dual fitting and primal fitting. In *WAOA*, pages 137–150, 2003. [2](#)

- [11] Nissan Lev-Tov and David Peleg. Polynomial time approximation schemes for base station coverage with minimum total radii. *Computer Networks*, 47(4):489–501, 2005. 2

A The Outer Cover Lower Bound

In this section, we provide the proof, adapted from [6], of [Theorem 1](#). For convenience, we restate the theorem.

Theorem 1. *Let $r' : Y \rightarrow \mathbb{R}^+$ be any assignment that constitutes a feasible solution to the MMC problem. For each $1 \leq i \leq k$, let μ_i denote the cost of an optimal outer cover of level i . Then*

$$\sum_{i=1}^k \mu_i \leq 3^\alpha \cdot \text{cost}(r').$$

Proof. Let $B = \{\delta(y, r'(y)) \mid y \in Y\}$ denote the set of balls corresponding to the assignment r' . We show that it is possible to form subsets $B_i \subseteq B$, for each $1 \leq i \leq k$ such that:

1. $\mu_i \leq 3^\alpha \cdot \text{cost}(B_i)$.
2. $B_i \cap B_j = \emptyset, \forall 1 \leq i \neq j \leq k$.
3. No two balls in B_i intersect, $\forall 1 \leq i \leq k$.

If we show this, [Theorem 1](#) follows because

$$\sum_{i=1}^k \mu_i \leq 3^\alpha \cdot \sum_{i=1}^k \text{cost}(B_i) \leq 3^\alpha \cdot \text{cost}(B) = 3^\alpha \cdot \text{cost}(r').$$

We create the set of balls B_i in a top-down manner as described in [Algorithm 2](#).

Algorithm 2 Compute-Balls

Require: The set of balls B corresponding to a k -cover assignment r'

Ensure: The set of balls $B_i, 1 \leq i \leq k$.

- 1: **for** $i = k$ to 1 **do**
 - 2: Let $\text{largest}^i(x) \leftarrow$ The largest ball in B that contains x .
 - 3: Let $B_i' = \{\text{largest}^i(x) \mid x \in X\}$.
 - 4: $B_i \leftarrow \emptyset$.
 - 5: **while** $B_i' \neq \emptyset$ **do**
 - 6: Let b be the largest ball in B_i' .
 - 7: $N \leftarrow$ Set of balls in B_i' that intersect b .
 - 8: $B_i \leftarrow B_i \cup \{b\}$.
 - 9: $B_i' \leftarrow B_i' \setminus N$.
 - 10: $B \leftarrow B \setminus B_i$.
-

We thus have a set of balls $B_i, 1 \leq i \leq k$. It is clear that $B_i \cap B_j = \emptyset$ (Property 2), and no two balls in B_i intersect (Property 3).

We now verify that each B_i also satisfies Property 1. For this, consider L_i , the set of balls obtained by increasing the radius of each ball in B_i by a factor of 3. We argue that L_i is an outer cover of level i for X .

Fix $x \in X$, and consider the ball $\text{largest}^i(x)$ in [Line 3](#) of iteration i . At this point, the balls in $\bigcup_{j=i+1}^k B_j$ have been removed from the original B , which had at least k balls containing x . Since no two balls in B_j intersect, there is at most one ball in each B_j that contains x . Thus, at this point, there are at least i balls left in B that contain x . Thus, the radius of $\text{largest}^i(x)$ is at least $d(x, y^i(x))$.

1. If $\text{largest}^i(x) \in B_i$, then the corresponding ball in L_i has radius at least $d(x, y^i(x))$.
2. If $\text{largest}^i(x) \notin B_i$, then there is an even larger ball b in B_i that intersects $\text{largest}^i(x)$. The ball obtained by multiplying the radius of b by 3 is in L_i ; it contains $\text{largest}^i(x)$ and thus x ; and it has radius at least $d(x, y^i(x))$.

Thus, L_i is an outer cover of level i for X . We infer that

$$\mu_i \leq \text{cost}(L_i) \leq 3^\alpha \cdot \text{cost}(B_i).$$

Thus, Property 1 holds. \square

B The Non-uniform MMC Problem

In this section, we address the non-uniform version of the metric multi-cover problem, which we refer to as the *non-uniform MMC*, and present an $O(1)$ approximation for it. Recall that the input consists of two point sets Y (servers) and X (clients) in an arbitrary metric space $(X \cup Y, d)$, a constant $\alpha \geq 1$, and a coverage function $\kappa : X \rightarrow \mathbb{Z}^+$.

Consider an assignment $r : Y \rightarrow \mathbb{R}^+$ of radii to each server in Y . We say that X is κ -covered under r (or by r) if for each $x \in X$, there are at least $\kappa(x)$ servers $y \in Y$ such that the ball of radius $r(y)$ centered at y contains x . That is, X is κ -covered if for each $x \in X$,

$$|\{y \in Y \mid d(x, y) \leq r(y)\}| \geq \kappa(x).$$

Any assignment $r : Y \rightarrow \mathbb{R}^+$ that κ -covers X is a feasible solution to the non-uniform MMC, and the goal is to find a feasible solution that minimizes the cost $\sum_{y \in Y} (r(y))^\alpha$. We assume that $\kappa(x) \leq |Y|$ for each client x , for otherwise there is no feasible solution.

Our algorithm for the non-uniform MMC extends that for the uniform case, and we adopt terminology from earlier. However, let k now denote $\max_{x \in X} \kappa(x)$. For client $x \in X$, let its set of *private servers* be $YP_x = YN_x(\lceil \kappa(x)/2 \rceil)$.

B.1 Filtering Clients

There is one obstacle that arises when attempting to extend [Algorithm 1](#) to the non-uniform MMC, and this motivates the following definition.

Definition B.1. *We say that client x_2 threatens client x_1 if*

- $\kappa(x_1) > \kappa(x_2)$, and
- $YN_{x_1}(\kappa(x_1) - \lfloor \kappa(x_2)/2 \rfloor) \cap YN_{x_2}(\kappa(x_2) - \lfloor \kappa(x_2)/2 \rfloor) \neq \emptyset$.

Observe that $YN_{x_2}(\kappa(x_2) - \lfloor \kappa(x_2)/2 \rfloor) = YP_{x_2}$, and thus the second condition informally says that some private servers of x_2 are also “inner” servers of x_1 .

To help understand the definition, consider the following example: suppose that $\kappa(x_1) = 100$, $\kappa(x_2) = 50$, and x_2 threatens x_1 . Thus, $YN_{x_1}(75) \cap YN_{x_2}(25) \neq \emptyset$. The plan for our algorithm is that it will provide the coverage required by x_2 in the first 25 iterations, and the coverage required by x_1 in the first 50 iterations. Now suppose that x_2 is chosen in the ruling set in the first 25 iterations. This precludes x_1 being in the ruling set in these first 25 iterations. However, we would like to allow x_1 to enter the ruling set in iteration 26, since x_2 is essentially finished at this point, whereas x_1 is not.

In each of the first 25 iterations, we would choose two servers for x_2 , one of which would be a server from $YN_{x_2}(25) = YP_{x_2}$. We would like at most one of these two servers to belong to $YN_{x_1}(75)$, so that x_1 has enough nearby servers when it later enters the ruling set. However, we

cannot ensure this, since the condition $YN_{x_1}(75) \cap YN_{x_2}(25) \neq \emptyset$ means that a private server of x_2 can belong to $YN_{x_1}(75)$.

As a preprocessing step, we compute a representative subset $\overline{X} \subseteq X$ in which no client threatens another:

Claim B.1. *We can compute in polynomial time a subset $\overline{X} \subseteq X$ of clients such that*

- *For any two clients x_1, x_2 such that x_2 threatens x_1 , $x_1 \in \overline{X} \implies x_2 \notin \overline{X}$;*
- *For any client $x \in X \setminus \overline{X}$, there is an $x' \in \overline{X}$ such that x threatens x' .*

Proof. Let ϕ be any ordering of the clients X such that the $\kappa(\cdot)$ values are non-increasing. Observe that if x_2 threatens x_1 then x_2 occurs after x_1 in ϕ . We initialize \overline{X} to be empty, and assume all clients are initially unmarked. We process each client in X according to the ordering ϕ as follows: for each client x , perform the following actions if x is unmarked: 1) add x to \overline{X} 2) mark all clients of X that threaten x .

It is easily checked that the resultant set of clients \overline{X} satisfies the two properties. \square

We will construct a hierarchy of ruling sets for clients \overline{X} . For any client $x \in X \setminus \overline{X}$, there is an $x' \in \overline{X}$ such that x threatens x' . Such an x' will help deal with the coverage requirements of x .

B.2 Family of Ruling Sets

For $1 \leq i \leq k$, we define the coverage function $\kappa_{-i} : X \rightarrow \mathbb{Z}^+$ by $\kappa_{-i}(x) = \max\{0, \kappa(x) - (i-1)\}$. Thus, κ_{-i} is obtained by decreasing the original coverage requirement of each client by $i-1$, with the proviso that we don't decrease below 0. For each $1 \leq i \leq k$, we define an undirected graph G_{-i} with vertex set \overline{X} . The pair (x, x') is an edge in G_{-i} if (a) $i \leq \lceil \kappa(x)/2 \rceil$; (b) $i \leq \lceil \kappa(x')/2 \rceil$; and (c) $YN_x(\kappa(x) - (i-1)) \cap YN_{x'}(\kappa(x') - (i-1)) \neq \emptyset$. Note that condition (a) and (b) hold, condition (c) can also be written as $YN_x(\kappa_{-i}(x)) \cap YN_{x'}(\kappa_{-i}(x')) \neq \emptyset$.

The conditions (a) and (b) ensure that a client x is isolated in graph G_{-i} (corresponding to the i -th iteration) for $i > \lceil \kappa(x)/2 \rceil$. This is in keeping with our vision that the first $i-1$ iterations already address the coverage for x by possibly using up servers close to x .

Observe that for $1 \leq j < i \leq k$, if (x, x') is an edge in G_{-i} it is also an edge in G_{-j} . Thus we may compute a family of ruling sets, obtaining an analog of [Claim 3.1](#).

Claim B.2. *There is a polynomial time algorithm that computes a hierarchy*

$$X_{-1} \subseteq X_{-2} \subseteq \dots \subseteq X_{-k},$$

where each $X_{-i} \subseteq \overline{X}$ is a $(3, 2)$ ruling set of G_{-i} .

B.3 The Main Algorithm

Our algorithm for the non-uniform MMC problem is described in [Algorithm 3](#). In many ways, it is analogous to [Algorithm 1](#), so we only highlight the key differences. One syntactic feature worth drawing attention to is that index i goes up from 1 in the for loop in [Line 4](#), as opposed to the for loop in [Line 4](#) of [Algorithm 1](#) where it decreased starting from k . Thus, iteration i in [Algorithm 3](#) corresponds to iteration $k - (i-1)$ in [Algorithm 1](#).

In iteration i , we consider each client $x_c \in X_{-i}$ in the for loop in [Line 6](#), but we add the farthest available server in $YN_{x_c}(\kappa(x_c) - (i-1))$ to Y_{-i}^s only if $\kappa(x) \geq 2i$, and any available server from YP_{x_c} to Y_{-i}^p only if $\kappa(x) \geq 2i-1$.

In the i -th iteration of the for loop in [Line 13](#), we use servers in Y_{-i}^s to 1-cover the clients with coverage demand at least $2i$, and Y_{-i}^p to 1-cover the clients with coverage demand at least $2i-1$. Notice that if k is odd and $i = l$, there are no clients with coverage demand at least $2i$.

Algorithm 3 NonUniformCover(X, Y, κ, α)

```
1: For each  $y \in Y$ , assign  $r(y) \leftarrow 0$  and mark  $y$  as available.
2:  $l \leftarrow \lceil k/2 \rceil$ 
3: Compute  $X_{-1} \subseteq X_{-2} \subseteq \dots \subseteq X_{-k}$  using Claim B.2.
4: for  $i = 1$  to  $l$  do
5:   Let  $Y_{-i}^s \leftarrow \emptyset, Y_{-i}^p \leftarrow \emptyset$ .
6:   for all  $x_c \in X_{-i}$  do
7:     if  $\kappa(x_c) \geq 2i$  then
8:        $y_s \leftarrow$  farthest available server in  $YN_{x_c}(\kappa(x_c) - (i - 1))$ .
9:        $Y_{-i}^s \leftarrow Y_{-i}^s \cup \{y_s\}$ . Mark  $y_s$  as not available.
10:    if  $\kappa(x_c) \geq 2i - 1$  then
11:       $y_p \leftarrow$  any available server in  $YP_{x_c}$ .
12:       $Y_{-i}^p \leftarrow Y_{-i}^p \cup \{y_p\}$ . Mark  $y_p$  as not available.
13: for  $i = 1$  to  $l$  do
14:   if  $k$  is even or  $i < l$  then
15:     Let  $r(Y_{-i}^s)$  be obtained by invoking Cover( $\cdot, Y_{-i}^s, \alpha$ ) for clients  $\{x \in X \mid \kappa(x) \geq 2i\}$ .
16:     Let  $r(Y_{-i}^p)$  be obtained by invoking Cover( $\cdot, Y_{-i}^p, \alpha$ ) for clients  $\{x \in X \mid \kappa(x) \geq 2i - 1\}$ .
17: return The assignment  $r : Y \rightarrow \mathbb{R}^+$ 
```

B.4 Server Availability

In this section, we show that Algorithm 3 finds available servers when it looks for them in Line 8 and Line 11. We define the *threshold level* of a client $x \in \overline{X}$ (denoted by $\text{th}(x)$) as the smallest i for which x belongs to the ruling set X_{-i} . (Some clients in \overline{X} may not be part of any of the ruling sets; when we refer to the threshold level of a client, we implicitly assume that it is in some ruling set, in particular, X_{-k} .) For client $x \in X$ and iteration $1 \leq i \leq \lceil \kappa(x)/2 \rceil$ of the for loop in Line 4, we define $A_x(i)$ to be the set of available servers within $YN_x(\kappa(x) - (i - 1))$ at the *beginning* of iteration i . Note that $A_x(1) = YN_x(\kappa(x))$.

To establish availability, it suffices to consider clients $x \in \overline{X}$ for which $\text{th}(x) \leq \lceil \kappa(x)/2 \rceil$. For a client $x \in \overline{X}$ for which $\text{th}(x) > \lceil \kappa(x)/2 \rceil$, the algorithm never looks for available servers in its neighborhood in Line 8 and Line 11.

We now show that any such client x has enough available servers at the beginning of iteration $i = \text{th}(x)$ of the outer loop of Algorithm 3. This argument is where the intricacies of the non-uniform MMC and the need for resolving “threats” show up.

Claim B.3. *Let x be any client in \overline{X} such that $\text{th}(x) \leq \lceil \kappa(x)/2 \rceil$, and let $i = \text{th}(x)$. Then*

$$(a) |A_x(i) \cap YP_x| \geq \lceil \kappa(x)/2 \rceil - (i - 1).$$

$$(b) |A_x(i)| \geq \kappa(x) - 2(i - 1).$$

Proof. Consider any iteration $j < i$ of the outer loop in Line 4. The client x itself is not part of the ruling set X_{-j} . Any client $x' \in X_{-j}$ for which some server is chosen in Line 8 or Line 11 must satisfy $j \leq \lceil \kappa(x')/2 \rceil$. For such a client x' , if $YN_x(\kappa(x) - (j - 1)) \cap YN_{x'}(\kappa(x') - (j - 1)) \neq \emptyset$, then (x, x') is an edge in G_{-j} . Since X_{-j} is a $(2, 3)$ -ruling set in G_{-j} , we conclude that there is at most one client $x' \in X_{-j}$ such that (a) some server is chosen in Line 8 or Line 11 for x' , and (b) $YN_x(\kappa(x) - (j - 1)) \cap YN_{x'}(\kappa(x') - (j - 1)) \neq \emptyset$. If there is no such client, we can conclude that in iteration j , no server in $YN_x(\kappa(x) - (j - 1))$ (and thus $YN_x(\kappa(x) - (i - 1))$) is made unavailable.

So let us assume that there is one such client x' . Next, we argue that $YN_x(\kappa(x) - (i - 1)) \cap YP_{x'} = \emptyset$. Since server choices are made for x' in iteration j , we have $j \leq \lceil \kappa(x')/2 \rceil$.

First consider the case $i \leq \lceil \kappa(x')/2 \rceil$. Since x and x' are both part of the ruling set X_{-i} , (x, x') is not an edge in G_{-i} . As $i \leq \lceil \kappa(x')/2 \rceil$ and $i \leq \lceil \kappa(x)/2 \rceil$, we may conclude

that $YN_x(\kappa(x) - (i - 1)) \cap YN_{x'}(\kappa(x') - (i - 1)) = \emptyset$. Also, since $i \leq \lceil \kappa(x')/2 \rceil$, we have $YP_{x'} \subseteq YN_{x'}(\kappa(x') - (i - 1))$. Thus, $YN_x(\kappa(x) - (i - 1)) \cap YP_{x'} = \emptyset$.

Next, consider the case $i > \lceil \kappa(x')/2 \rceil$. Since $\lceil \kappa(x)/2 \rceil \geq i$, we have that $\kappa(x) > \kappa(x')$. Now, since x' does not threaten x , we conclude that $YN_x(\kappa(x) - \lfloor \kappa(x')/2 \rfloor) \cap YN_{x'}(\kappa(x') - \lfloor \kappa(x')/2 \rfloor) = \emptyset$. Since $YN_x(\kappa(x) - (i - 1)) \subseteq YN_x(\kappa(x) - \lfloor \kappa(x')/2 \rfloor)$, and $YN_{x'}(\kappa(x') - \lfloor \kappa(x')/2 \rfloor) = YP_{x'}$, we conclude that $YN_x(\kappa(x) - (i - 1)) \cap YP_{x'} = \emptyset$.

Thus, in iteration j , the server choice made for x' in [Line 11](#) is not from $YN_x(\kappa(x) - (i - 1))$, whereas the server choice made for x' in [Line 8](#) may be from $YN_x(\kappa(x) - (i - 1))$.

Since at most one server from $YN_x(\kappa(x) - (i - 1))$ is made unavailable in each of the $i - 1$ iterations before iteration i , we conclude that $A_i(x) \geq \kappa(x) - (i - 1) - (i - 1)$. The first assertion of the lemma also follows. \square

The next claim says that before every iteration $\text{th}(x) \leq i \leq \lceil \kappa(x)/2 \rceil$, there are enough available servers in $YN_x(\kappa(x) - (i - 1))$. These are iterations in which x itself is part of the ruling set, and the argument is identical to that of [Claim 3.6](#).

Claim B.4. *Let x be any ruling client and $\text{th}(x) \leq i < \lceil \kappa(x)/2 \rceil$. Then*

- (a) $|A_x(i + 1)| \geq |A_x(i)| - 2$
- (b) *If $|A_x(i + 1)| = |A_x(i)| - 2$, then one of the servers in $A_x(i) \setminus A_x(i + 1)$ is the farthest server in $A_x(i)$ from x .*

We can now assert our final claim about server availability. The proof follows from [Claim B.3](#) and [Claim B.4](#) using arguments very similar to [Claim 3.7](#).

Claim B.5. *Algorithm 3 finds an available server whenever it executes [Line 11](#) or [Line 8](#).*

B.5 Outer Covers

We generalize the notion of an outer cover as used in the uniform MMC. Let $\kappa' : X \rightarrow \mathbb{Z}^+$ be a coverage function where as usual we assume $\kappa'(x) \leq |Y|$ for any client x . For notational convenience, we denote $y^{\kappa'(x)}(x)$, the $\kappa'(x)$ -th nearest server of client x , by $\text{nn}(\kappa', x)$.

A κ' -outer cover is an assignment $\rho : Y \rightarrow \mathbb{R}^+$ of radii to the servers such that for each client $x \in X$ for which $\kappa'(x) > 0$, there is a server $y \in Y$ such that

1. The ball $\delta(y, \rho(y))$ contains x i.e. $d(y, x) \leq \rho(y)$.
2. Radius of the ball at y is large, that is, $\rho(y) \geq d(x, \text{nn}(\kappa', x))$

Given a level κ' -outer cover ρ , and a client $x \in X$ with $\kappa'(x) > 0$, any server y that satisfies the two conditions in the definition above is said to *serve* x ; we also say that the corresponding ball $\delta(y, \rho(y))$ serves x . Observe that we do not require that a client x with $\kappa'(x) = 0$ be covered or served. Also observe that an outer cover of level i is a special case of a κ' -outer cover where κ' is the constant function that takes on the value i .

The following lemma gives a lower bound on the cost of the optimal solution for the non-uniform MMC. It is analogous to [Theorem 1](#) and its proof follows by similar arguments. Recall that for $1 \leq i \leq k$, the coverage function $\kappa_{-i} : X \rightarrow \mathbb{Z}^+$ is defined by $\kappa_{-i}(x) = \max\{0, \kappa(x) - (i - 1)\}$.

Lemma 3. *Let $r' : Y \rightarrow \mathbb{R}^+$ be any assignment that constitutes a feasible solution to the non-uniform MMC. For each $1 \leq i \leq k$, let μ_{-i} denote the cost of an optimal κ_{-i} -outer cover. Then*

$$\sum_{i=1}^k \mu_{-i} \leq 3^\alpha \cdot \text{cost}(r').$$

B.6 Approximation Guarantee

To obtain an approximation guarantee for [Algorithm 3](#), we first upper bound the cost of the covers returned in iteration i of the for loop in [Line 13](#).

Claim B.6. *Assume that either (a) k is even and $1 \leq i \leq l$, or (b) k is odd and $1 \leq i < l$. Let ρ^{-i} be any κ_{-i} -outer cover. There is a 1-cover of the clients $\{x \in X \mid \kappa(x) \geq 2i\}$ that uses servers from Y_{-i}^s and has cost at most $16^\alpha \cdot \text{cost}(\rho^{-i})$.*

Proof. Let $Z = \{x \in X \mid \kappa(x) \geq 2i\}$. The proof is similar to that of [Claim 3.9](#), but there are two additional issues to deal with. The first is that we have to address clients in Z that do not belong to \overline{X} . The second is that for each client in Z we have to find a “nearby” client in the ruling set $\bar{x} \in X_{-i}$ such that $\kappa(\bar{x}) \geq 2i$. It does not suffice if $\kappa(\bar{x}) = 2i - 1$, because such a client does not add a neighboring server to Y_{-i}^s .

Consider the set B of balls obtained by expanding each ball in the outer cover ρ^{-i} to 8 times its original radius. It suffices, as in the proof of [Claim 3.9](#), to show the following claim.

Claim B.7. *For any client $x \in Z$, there is some ball in B that contains x as well as at least one server in Y_{-i}^s .*

We now turn to the proof of [Claim B.7](#). Let us first consider the case where $x \in Z \cap \overline{X}$. Since X_{-i} is a (2,3)-ruling set in G_{-i} , there is a path π in G_{-i} with at most 2 edges (and 3 vertices) that connects x to some vertex $\bar{x} \in X_i$. Let $\delta(y, \rho^{-i}(y))$ be the biggest ball in outer cover ρ^{-i} that serves at least one vertex on path π . Suppose that it serves vertex $\hat{x} \in \pi$. (\hat{x} could be the same as x or \bar{x} .) Note that vertices x' in G_{-i} with $i > \lceil \kappa(x')/2 \rceil$ are isolated. Thus, $\kappa(x') \geq 2i - 1$ for any vertex x' on this path. We claim that in fact $\kappa(x') \geq 2i$ for any vertex x' . Otherwise, since $\kappa(x) \geq 2i$, there is an edge (x', x'') in π such that $\kappa(x') = 2i - 1$, and $\kappa(x'') \geq 2i$. Since (x', x'') is an edge in G_{-i} , we have

$$YN_{x'}(\kappa(x') - (i - 1)) \cap YN_{x''}(\kappa(x'') - (i - 1)) \neq \emptyset.$$

As $i - 1 = \lfloor \kappa(x')/2 \rfloor$, we see that x' threatens x , a contradiction. We conclude that $\kappa(x') \geq 2i$ for any vertex x' on π .

Using the definition of a κ_{-i} , and the way we pick the ball $\delta(y, \rho^{-i}(y))$, it follows that for any vertex $x' \in \pi$,

$$d(x', \text{nn}(\kappa_{-i}, x')) \leq \rho^{-i}(y). \quad (3)$$

From the definition of G_{-i} , we have that for any edge (x', x'') in π ,

$$d(x', x'') \leq d(x', \text{nn}(\kappa_{-i}, x')) + d(x'', \text{nn}(\kappa_{-i}, x'')) \leq 2\rho^{-i}(y). \quad (4)$$

Thus, using [Inequality \(4\)](#),

$$d(y, x) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, x]} d(x', x'') \right) \leq 5\rho^{-i}(y).$$

Now, because $\kappa(\bar{x}) \geq 2i$, we add a server from $YN_{\bar{x}}(\kappa(\bar{x}) - (i - 1))$ to Y_{-i}^s . Let \bar{y} be this server. Clearly, $d(\bar{x}, \bar{y}) \leq d(\bar{x}, \text{nn}(\kappa_{-i}, \bar{x})) \leq \rho^{-i}(y)$.

We calculate that

$$d(y, \bar{y}) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi[\hat{x}, \bar{x}]} d(x', x'') \right) + d(\bar{x}, \bar{y}) \leq 6\rho^{-i}(y).$$

Thus, the ball $\delta(y, 6\rho^{-i}(y))$ contains both x and $\bar{y} \in Y_{-i}$, completing the proof of [Claim B.7](#) for the clients in $Z \cap \overline{X}$.

Now consider an arbitrary client $x_1 \in Z \setminus \overline{X}$. There is a client $x \in \overline{X}$ such that x_1 threatens x . Thus, $\kappa(x) \geq \kappa(x_1)$, so $x \in Z \cap \overline{X}$. Furthermore,

$$YN_x(\kappa(x) - \lfloor \kappa(x_1)/2 \rfloor) \cap YN_{x_1}(\kappa(x_1) - \lfloor \kappa(x_1)/2 \rfloor) \neq \emptyset.$$

Since $i - 1 \leq \lfloor \kappa(x_1)/2 \rfloor$, we have

$$YN_x(\kappa(x) - (i - 1)) \cap YN_{x_1}(\kappa(x_1) - (i - 1)) \neq \emptyset.$$

We conclude that

$$d(x, x_1) \leq d(x, \mathbf{nn}(\kappa_{-i}, x)) + d(x_1, \mathbf{nn}(\kappa_{-i}, x_1)). \quad (5)$$

Now, let π be a path in G_{-i} with at most two edges that connects x to $\bar{x} \in X_{-i}$. Let $\delta(y, \rho^{-i}(y))$ be the biggest ball in outer cover ρ^{-i} that serves at least one of $\{x_1\} \cup \{x' \mid x' \in \pi\}$. Suppose it serves vertex \hat{x} .

Using [Inequality \(5\)](#), we have $d(x, x_1) \leq 2\rho^{-i}(y)$. Let π' be the path obtained by concatenating (x_1, x) to π . (π' is not a path in G_{-i} .) Let \bar{y} be the server from $YN_{\bar{x}}(\kappa(\bar{x}) - (i - 1))$ that is added to Y_{-i}^s .

We calculate

$$d(y, x_1) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi'[\hat{x}, x_1]} d(x', x'') \right) \leq 7\rho^{-i}(y),$$

and

$$d(y, \bar{y}) \leq d(y, \hat{x}) + \left(\sum_{(x', x'') \in \pi'[\hat{x}, \bar{x}]} d(x', x'') \right) + d(\bar{x}, \bar{y}) \leq 8\rho^{-i}(y).$$

Thus, the ball $\delta(y, 8\rho^{-i}(y))$ contains both x_1 and $\bar{y} \in Y_{-i}$, completing the proof of [Claim B.7](#). \square

The following claim addresses the cost of the cover obtained using the server set Y_{-i}^p . Its proof is very similar to that of [Claim B.6](#). It is simpler, because one of the two issues mentioned in the earlier proof does not have to be handled.

Claim B.8. *Let $1 \leq i \leq l$ and ρ^{-i} be any κ_{-i} -outer cover. There is a 1-cover of the clients $\{x \in X \mid \kappa(x) \geq 2i - 1\}$ that uses servers from Y_{-i}^p and has cost at most $16^\alpha \cdot \text{cost}(\rho^{-i})$.*

We can now establish the approximation guarantee for [Algorithm 3](#) and the main result of this section.

Theorem 4. *Given point sets X and Y in a metric space $(X \cup Y, d)$ and a coverage function κ , [Algorithm 3](#) runs in polynomial time and returns a κ -cover of X with cost at most $2 \cdot (16 \cdot 9)^\alpha$ times that of an optimal κ -cover.*

Proof. It is evident that the algorithm runs in polynomial time. It is also easy to check that the assignment r that it returns is a κ -cover, that is, each client x is covered at least $\kappa(x)$ times. Let r' be any optimal κ -cover. By [Lemma 3](#), there exists a κ_{-i} -outer cover ρ^{-i} , for $1 \leq i \leq k$ such that

$$\sum_{i=1}^k \text{cost}(\rho^{-i}) \leq 3^\alpha \text{cost}(r').$$

From [Claim B.6](#) and [Claim B.8](#), and the fact that $\text{Cover}(\cdot, \cdot, \alpha)$ returns a 3^α approximation, we conclude that the cost of a 1-cover that is computed in iteration i of the for loop in [Line 13](#) is at most $(16 \cdot 3)^\alpha \text{cost}(\rho^{-i})$. At most two 1-covers are computed in iteration i . Thus,

$$\text{cost}(r) \leq 2 \cdot (16 \cdot 3)^\alpha \cdot \sum_{i=1}^l \text{cost}(\rho^{-i}) \leq 2 \cdot (16 \cdot 3)^\alpha \cdot \sum_{i=1}^k \text{cost}(\rho^{-i}) \leq 2 \cdot (16 \cdot 9)^\alpha \cdot \text{cost}(r').$$

□